An exemplary input message supported by DirLister class 172 is:

<message name="DirLister_REQUEST">
<path>path name of directory</path>
<short>true</short>
<long>false</long>
<info>false</info>
<deep>false</deep>
</message>

The field name "message" is in the root element node of the first DOM document. In addition, the attribute name "name" and the attribute value "DirLister_REQUEST" are also included in the root element node of the first DOM document to indicate the subclass of BusinessService class 48 is DirLister class 172. The field name "path" includes a string indicating the path name of the directory of interest, such as, for example, "C:\my documents." The field name "short" includes a Boolean function indicating file names are desired. The field name "long" includes another Boolean function indicating the path for each file is not desired. The field name "info" is another Boolean function indicating the file size and modification date are not desired. The field name "deep" is another Boolean function indicating recursive retrieval of data for files within subdirectories is not desired.

The corresponding Message/Field object structure is:

Message "DirLister_REQUEST"
!
+---Field "path"
+---Field "short"
+---Field "long"
+---Field "info"
+---Field "deep"

The representative first DOM document is:

!
+---Element "message"
    ! !
    ! +---Attributes "name=DirLister_REQUEST"

```
!
+---Element "path"
! !
! +---TextNode "pathname of directory"
!
+---Element "short"
! !
! +---TextNode "true"
!
+---Element "long"
! !
! +---TextNode "false"
!
+---Element "info"
! !
! +---TextNode "false"
!
+---Element "deep"
 !
 +---TextNode "false"
```

Within an instance of DirLister class 172, the request parameters from the input message are used by the custom application code to extract data from the datafile 178. The units of data within the input message for those fields not common to all input messages are defined by DirLister_Request class 174. As previously discussed, MESSAGEDEFINITION class 50 includes the field names and indicates the expected datatype for those fields common to all input messages. DirLister_Request class 174 operates similarly and includes those fields specific to requests directed to DirLister class 172.

Depending on the status of the mode debug flag, DirLister class 172 may use either shortname or longname as the field names. The shortname and longname field names are defined in DirLister_Request class 174 and MESSAGEDEFINITION class 50. For example, if the field name "path" is the longname and "b1" is the short name,

the above input message could have the fieldname translated to "b1" to facilitate efficient transmission to DirLister class 172.

As a function of the request parameters, DirLister class 172 generates a response that is hierarchical representation of the directory contents. The response is translated to an output message using definitions for the units of data and shortname and longname field names from DirLister_Reply class 176 and MESSAGEDEFINITION class 50. An exemplary output message is:

<message name="DirLister_REPLY">
<directory shortname="Sample">
<file shortname="patent.doc"/>
<file shortname="pictures.ppt"/>
<directory shortname="source" unexpanded="true"/>
</directory>
</message>

To generate the output message, the serviceMain method calls a traverse method of DirLister class 172. The traverse method determines what the "path" field name is associated with. If the data is identified as a directory, the directory is traversed by the traverse method. Conversely, if the data is a file, the desired data is extracted as a function of the selected request parameters. To traverse the directory, a traverse2 method within DirLister class 172 is initialized. The traverse2 method recursively traverses the directory tree.

In one embodiment, both the traverse method and the traverse2 method use Java APIs and associated classes to read in the data and form a response. Operation of the Java APIs and associated classes of the traverse and traverse2 methods are unassociated with the operation of the core classes within the business services layer 16. As such, the traverse and traverse2 methods may be built to operate within the framework provided by the business services layer 16 without affecting the functionality of the core classes within the business services layer 16.

For every directory entry read, a createEntry method of DirLister class 172 is called to generate the appropriate field and attributes in the output message. The createEntry method calls the createField method of Message class 44 to add fields to the second DOM document. In addition, the createEntry method calls the setAttribute